# Avaliação da aplicação do balanceador de carga GROUPLB em aplicações iterativas *

**Giovane Lizot[1], Anderson Felipe Ribeiro[1], Edson L. Padoin[1]**

[1]Universidade Reg. do Noroeste do Estado do Rio G. do Sul (UNIJUI) - Ijuí - RS - Brasil

giovanerosalizott@hotmail.com, anderson.ribeiro.r@outlook.com

padoin@unijui.edu.br

***Resumo.*** *Este artigo apresenta os resultados da aplicação de uma nova versão do balanceador de carga* GROUPLB *em um benchmark e uma aplicação real. O seu objetivo é analisar os ganhos de desempenho alcançados em relação à outros balanceadores de carga do estado da arte. O Balanceador de Carga foi implementado utilizando o modelo de programação* CHARM++ *buscando a redução do tempo de execução de aplicações iterativas executadas em ambientes paralelos de memória compartilhada. Os resultados iniciais demonstram reduções no tempo total de execução em comparação com a abordagem sem balanceador. Os resultados iniciais apresentam ganhos de até 29,6% quando aplicado na execução do benchmark lb_test de até 22,5% quando aplicado na execução da aplicação Lassen.*

## 1. Introdução

O crescimento da computação de alto desempenho, ocorrido pelo avanço novas tecnologias, tem permitido simulações de problemas cada vez mais complexos. No entanto, muitas destas simulações apresentam desequilíbrio de carga quando executadas em ambientes paralelos. Outras aplicações, por sua vez, quando paralelizadas apresentam comunicação excessiva entre suas tarefas impedindo a utilização eficiente dos recursos paralelos. Desta forma, nodos computacionais podem receber tarefas com menor carga e permanecer ociosos enquanto outros executam grandes tarefas, reduzindo assim o desempenho da aplicação [Padoin et al. 2014].

Atualmente os grandes sistemas computacionais possuem múltiplas unidades de processamento paralelo. Assim sendo, a paralelização eficiente das aplicações tornou-se necessária de modo que explore todos os recursos paralelos. Nesse contexto, a detecção de desequilíbrios de cargas, o emprego de estratégias de tomadas de decisão para a migração de tarefas faz-se cada vez mais necessário para aumentar a eficiência dos sistemas. A grande maioria das simulações que são executadas em ambientes paralelos apresentam comportamentos dinâmicos devido aos cálculos baseados em fórmulas complexas. Tal comportamento sugere o uso de estratégias de balanceamento de carga para melhorar o equilíbrio de carga e reduzir a quantidade de comunicações. Assim, almejando mitigar estes problemas, balanceadores de carga estão sendo propostos e cada vez mais são utilizados em simulações para aumentar a eficiência de utilização dos recursos paralelos dos atuais sistemas computacionais [da Rosa Lizot et al. 2018].

O restante do trabalho está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta a proposta do balanceador de carga GROUPLB. A

*Trabalho desenvolvido com recursos do edital MCTIC/CNPq - Universal 28/2018 sob número 436339/2018-8 e do edital da VRPGPE bolsa PIBIC/UNIJUI.

Seção 4 descreve a metodologia utilizada na implementação e o ambiente de execução utilizado na realização dos testes. Resultados são discutidos na Seção 5, seguidos das conclusões e trabalhos futuros.

## 2. Trabalhos Relacionados

A computação paralela tem ganhado espaço tanto em plataformas com memória compartilhada quanto em memória distribuída. No entanto, muitas aplicações paralelas apresentam tarefas com diferentes demandas de processamento em suas tarefas gerando desequilíbrio de carga. Para isto, diferentes abordagens de balanceamento de carga tem sido desenvolvidas. Dentre elas destacam-se as *centralizadas* [Bhatelé et al. 2008] e as *distribuídas* [Zheng et al. 2011]. Por outro lado, abordagens *hierárquicas* também vem sendo propostas almejando reduzir a sobrecarga das demais abordagens [Zheng et al. 2011].

Uma característica que diferencia estas abordagens é a quantidade de nodos computacionais utilizada para a tomada de decisão do balanceamento de carga. Nas estratégias *centralizadas* a tomada de decisão de balanceamento de carga ocorre em um único nodo. Para isso, os dados de carga e comunicação de todas as tarefas são acumulados em um nodo específico, que executa um processo de decisão.

Os balanceadores de carga GREEDYLB e REFINELB são os mais utilizados nesse tipo de estratégia. GREEDYLB usa um algoritmo de abordagem gulosa que implementa práticas de otimização combinatória. Seu algoritmo, migra objetos pesados para o processador com menos carga. Isso é repetido até que a carga de todos os nodos computacionais alcance uma carga média. Diferentemente, REFINELB usa uma abordagem baseada no refinamento de carga. Esse balanceador de carga move as tarefas dos nodos computacionais mais sobrecarregados para os menos carregados até atingir uma média utilizando um limite no número de objetos que podem ser migrados [Freytag et al. 2015]. Os balanceadores AVERAGELB [Arruda et al. 2015] e SMARTLB [dos Santos et al. 2018] também adotam abordagem centralizada em suas estratégias de tomada de decisões.

Para a implementação da estratégia proposta considerou-se os balanceadores de carga GREEDYLB e REFINELB do estado da arte, e AVERAGELB [Arruda et al. 2015], e SMARTLB [dos Santos et al. 2018].

## 3. Balanceador de carga GROUPLB

O balanceador de carga proposto amplia os mecanismos de tomada decisões dos balanceadores AVERAGELB e SMARTLB. Utilizando também uma abordagem centralizada em sua estratégia, coleta informações de carga e toma suas decisões em um único nodo. A estratégia leva em consideração a média aritmética da soma das cargas dos nodos do processador, buscando um equilíbrio entre as cargas e uma redução do número de migrações. A estratégia é realizada em 4 etapas:

- Na **etapa 1** os processos são mapeados de acordo com as suas cargas computacionais, calculando a média total de cada nodo.

- Na **etapa 2** é realizado um somatório das cargas dos processos de cada grupo. Para a tomada de decisões, o algoritmo utiliza 3 variáveis de controle. Variável *pp* com a soma das cargas dos processos do menor grupo, ou seja, o do *Small*. Variável *gg* com a soma das cargas dos processos do grupo *Large* e Variável *delta* com a diferença entre as variáveis pp e gg, ou seja, o desbalanceamento entre os grupos *Small* e *Large*.

- Na **etapa 3** inicia-se a migração. O algoritmo leva em consideração a carga total de cada grupo, a média total e o delta e move os processos entre os grupos *Small* e *Large*, para que a diferença entre os grupos diminua.

Na **etapa 4** o algoritmo move os processos considerando o valor da média, os menores que a média para o grupo *Large*, e os maiores para o grupo *Small*.

Após cada migração um novo delta é calculado. Assim, as cargas tendem a ser distribuídas entre os cores de acordo com a estratégia adotada, reduzindo o desequilíbrio. Portanto, migrações desnecessárias não são executadas reduzindo o tempo de execução do aplicativo, e a cada iteração fazendo uma varredura das cargas nos cores e migrando de acordo com a necessidade.

Para a implementação do balanceador de carga proposto, foi selecionado o ambiente CHARM++. Ehttps://www.overleaf.com/project/5db05ddb6c50110001bfadd8sse ambiente é suportado por várias plataformas, permitindo que os programas desenvolvidos neste modelo sejam executados em ambientes de memória compartilhada e distribuída.

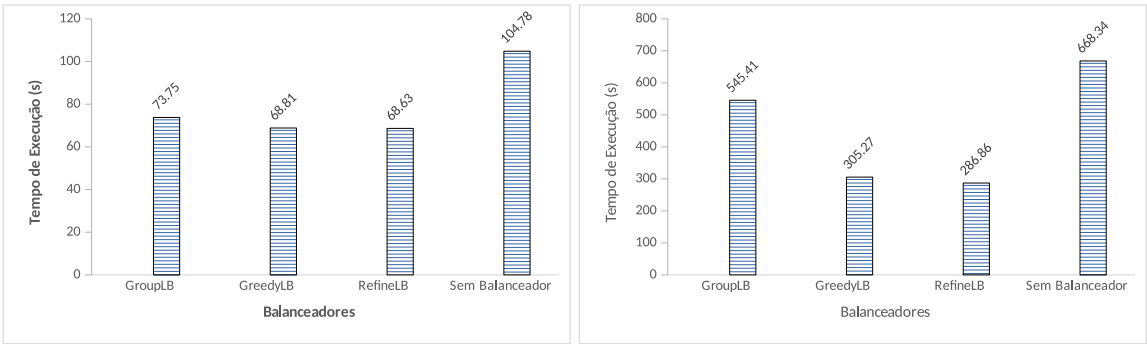
## 4. Metodologia

Para validar a estratégia de proposta foi utilizado um equipamento com processador Intel Core i7-8700 com 6 núcleos físicos. Para os testes, foi utilizado o sistema operacional Linux Ubuntu 18.04 com kernel versão 4.4.33-1. A versão do Charm ++ usada foi 6.10 e o compilador g++ na versão 6.2.1. Cada um dos testes realizados neste trabalho foi repetido 10 vezes.

Para avaliar o desempenho do balanceador de carga proposto, ele foi aplicado na execução do benchmark *lb_test* e da aplicação *Lassen* comparado com a abordagem sem balanceador e também aos outros dois balanceadores, GREEDYLB e REFINELB, fornecidos pelo ambiente de programação CHARM++. Os testes foram realizados com 300 tarefas, sendo estas cargas computacionais variando entre 1500 ms e 200000 ms, para o Benchmark *lb_test*. Para a aplicação *Lassen* foram testadas tarefas com a configuração básica. Sincronizações para chamada do balanceador foram definidas a cada 10 iterações.

## 5. Resultados

Na Figura 1 são apresentados os tempos mensurados nas execuções realizadas com o benchmark *lb_test* e a aplicação *Lassen* utilizando diferentes balanceadores de carga.




(a) *lb_test* (b) *Lassen*

**Figura 1. Tempos de execução mensurados durante os testes.**

Nos testes realizados com o benchmark *lb_test* com $300$ tarefas, o balanceador GROUPLB obteve ganhos significativos. O tempo total de execução foi reduzido de $104,78$ segundos da execução sem balanceador para $73,74$ segundos, com ganhos de até $29,62\%$. O balanceador de carga GROUPLB também reduziu o tempo quando aplicado na execução da aplicação Lassen. Nos testes o tempo foi reduzido de $668,34$ para $545,41$ segundos, o que representa um ganho de $22,53\%$.

Os resultados desta nova versão do GROUPLB não foram tão significativos quanto aos ganhos dos balanceadores GREEDYLB e REFINELB, estes já consolidados na literatura.

## 6. Conclusões e trabalhos futuros

Este trabalho apresentou uma análise de desempenho de uma nova proposta do balanceador de carga GROUPLB. Os resultados preliminares apresentaram ganhos, tendo o algoritmo proposto reduzido o tempo de execução com ganhos superiores a abordagem sem balanceador de carga. Percebeu-se também uma redução da quantidade de migração de tarefas nos testes realizados. No entanto, a proposta apresenta resultados inferiores aos balanceadores de carga GREEDYLB e REFINELB já consolidadas.

Como futuros trabalhos, pretende-se continuar implementando melhorias no algoritmo do GROUPLB durante a tomada de decisão sobre a ordenação das tarefas. Também pretende-se analisar os ganhos alcançados considerando outros sistemas paralelos, bem como outros benchmarks e aplicações reais de computação científica.

## Referências


Arruda, G., Padoin, E. L., Pilla, L. L., Navaux, P. O. A., and Mehaut, J.-F. (2015). Proposta de balanceamento de carga para redução de migração de processos em ambientes multiprogramados. In *XVI Simpósio de Sistemas Computacionais (WSCAD-WIC)*, pages 1–8.

Bhatelé, A., Kumar, S., Mei, C., Phillips, J. C., Zheng, G., and Kalé, L. V. (2008). Overcoming scaling challenges in biomolecular simulations across multiple platforms. In *Proceedings...*, pages 1–12. International Symposium on Parallel and Distributed Processing (IPDPS), IEEE.

da Rosa Lizot, G., Mastella, V. M., Pavan, P. J., and Padoin, E. L. (2018). Grouplb: Load balancer proposal to reduce the execution time of parallel applications. In *XVI Parallel and Distributed Processing Workshop (WSPPD)*, pages 1–4, Porto Alegre, RS.

dos Santos, V. R. S., Padoin, E. L., Navaux, P. O. A., and Méhaut, J.-F. (2018). Smartlb: Proposta de um balanceador de carga para redução de tempo de execução de aplicações em ambientes paralelos. In *WPERFORMANCE*, pages 1–14.

Freytag, G., Arruda, G., Martins, R. S. M., and Padoin, E. L. (2015). Análise de desempenho da paralelização do problema de caixeiro viajante. In *XV Escola Regional de Alto Desempenho (ERAD)*, pages 1–4, Gramado, RS. SBC.

Padoin, E. L., Castro, M., Pilla, L. L., Navaux, P. O., and Méhaut, J.-F. (2014). Saving energy by exploiting residual imbalances on iterative applications. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10. IEEE.

Zheng, G., Bhatelé, A., Meneses, E., and Kalé, L. V. (2011). Periodic hierarchical load balancing for large supercomputers. *International Journal of High Performance Computing Applications*, 25(4):371–385.